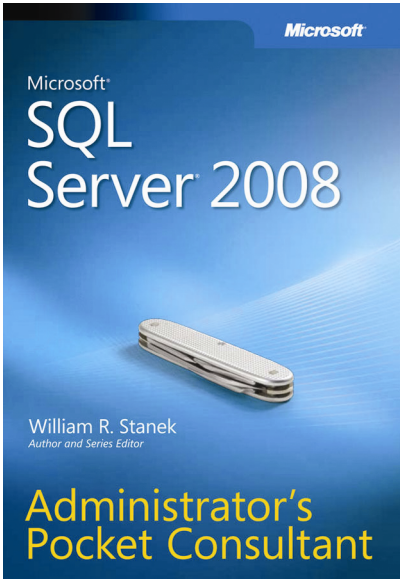

Microsoft® SQL Server® 2008 Administrator's Pocket Consultant

William R. Stanek

To learn more about this book, visit Microsoft Learning at http://www.microsoft.com/MSPress/books/12755.aspx

9780735625891

Microsoft® Press

Table of Contents

Chapter 5

Managing the Enterprise

In this chapter:

Microsoft SQL Server Management Studio is the primary tool you will use to manage database servers. Other tools that are available to manage local and remote servers include SQL Server Configuration Manager, Reliability And Performance Monitor, and Event Viewer. You will use SQL Server Configuration Manager to manage SQL Server services, networking, and client configurations. Reliability And Performance Monitor is available to track SQL Server activity and performance, and Event Viewer provides a way to examine events generated by SQL Server, which can provide helpful details for troubleshooting. In this chapter, you will learn how to use SQL Server Management Studio. SQL Server Configuration Manager is discussed in Chapter 3, "Managing the Surface Security, Access, and Network Configuration." For details on Performance Monitor and Event Viewer, see Chapter 14, "Profiling and Monitoring Microsoft SQL Server 2008."

Using SQL Server Management Studio

The SQL Server Management Studio graphical point-and-click interface makes server, database, and resource management easy to do. Using SQL Server Management Studio, you can manage both local and remote server instances by establishing a connection to SQL Server and then administering its resources. If you have disabled remote server connections to a particular server, however, you can work only with the server locally (by logging on to the system at the keyboard or by establishing a remote Terminal Server session in Microsoft Windows and then running the local management tools).

Getting Started with SQL Server Management Studio

To run SQL Server Management Studio, click the Start button, point to All Programs, Microsoft SQL Server 2008, and then select SQL Server Management Studio. Then you must connect to the server with which you want to work. There are several ways to do this:

- Connect using a standard login to a server instance.
- Connect using a login to a specific database.
- Connect using server groups and registered servers.

Connecting to a server instance allows you to work with that particular server and its related components. Typically, you will want to connect to a server's Database Engine. As you can see in Figure 5-1, the Database Engine gives you access to the following components and features:

- **Databases** Manage system databases, including *master* and *model*, as well as user databases and database snapshots. You can also access the ReportServer and Report ServerTempDB databases under this node.
- **Security** Manage SQL logins, server roles, stored credentials, cryptographic providers, and auditing.
- **Server Objects** Configure backup devices, HTTP endpoints, linked servers, and server triggers.
- **Replication** Configure distribution, update replication passwords, and launch Replication Monitor.
- **Management** Configure SQL Server logs, maintenance plans, Distributed Transaction Coordinator, and Database Mail. Configure data collection, Resource Governor, and Policy-Based Management policies. You can also configure legacy features, such as SQL Server 2000 database maintenance plans, SQL Mail, and DTS 2000 packages.
- **SQL Server Agent** Configure SQL Server Agent jobs, alerts, operators, proxies, and error logs.

If you are not automatically connected or you exited the Connect To Server dialog box, you can connect to a server instance by clicking Connect in the Object Explorer view. You store server and login information using the registration feature. Registered servers can be organized using server groups and then can be accessed quickly in the Registered Servers view. Methods to manage server groups and register servers are discussed in "Managing SQL Server Groups" on page 116 and "Managing Servers" on page 119, respectively.

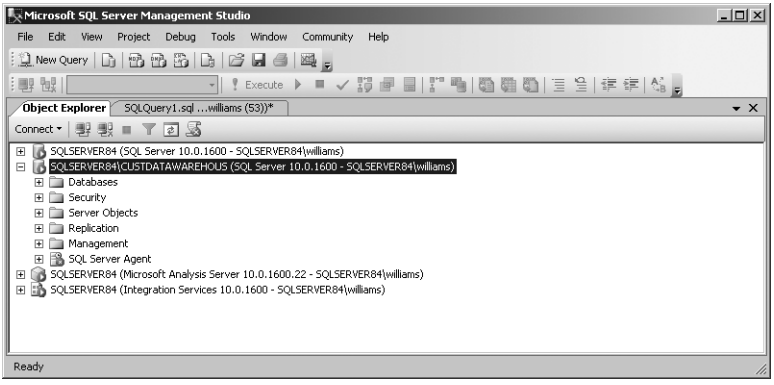

Figure 5-1 Use the Database Engine to access core SQL Server components and features.

Connecting to a Specific Server Instance

To connect to a specific server instance using a standard logon, follow these steps:

1. Start SQL Server Management Studio by clicking the Start button, pointing to All Programs, Microsoft SQL Server 2008, and then selecting SQL Server Management Studio.
2. In the Connect To Server dialog box, use the Server Type drop-down list to select the database component to which you want to connect, such as Database Engine.
3. In the Server Name box, type the fully qualified or host name of the server on which SQL Server is running, such as **corpsvr04.cpandl.com** or **CorpSvr04**. Or select Browse For More on the related drop-down list. In the Browse For Server dialog box, select the Local Servers or Network Servers tab as appropriate. After the instance data has been retrieved, expand the nodes provided, select the server instance, and then click OK.

   Note The list in the Browse For Servers dialog box is populated by the SQL Server Browser service running on the database servers. There are several reasons why a SQL Server instance you want to work with might not be listed. The SQL Server Browser service might not be running on the computer running SQL Server. A firewall might be blocking UDP port 1434, which is required for browsing. Or the HideInstance flag might be set on the SQL Server instance.

4. Use the Authentication drop-down list to choose the authentication type, either Windows Authentication or SQL Server Authentication (based on the authentication types selected when you installed the server). Provide a Windows user name or SQL Server login ID and password as necessary.

- **Windows Authentication** Uses your current domain account and password to establish the database connection. This authentication type works only if Windows authentication is enabled and you have appropriate privileges.
- **SQL Server Authentication** Allows you to specify a SQL Server login ID and password. To save the password so that you do not have to re-enter it each time you connect, select Remember Password.

5. Click Connect. You can now use the Object Explorer view to work with this server.

Connecting to a Specific Database

To connect to a specific database using a standard login, follow these steps:

1. Start SQL Server Management Studio by clicking the Start button, pointing to All Programs, Microsoft SQL Server 2008, and then selecting SQL Server Management Studio. Or click Start, type **ssms.exe** in the Search box, and then press Enter.
2. In the Connect To Server dialog box, use the Server Type drop-down list to select the database component to which you want to connect, such as Database Engine, and then, in the Server Name box, type the fully qualified or host name of the server on which SQL Server is running, such as **corpsvr04.cpandl.com** or **CorpSvr04**.
3. Use the Authentication drop-down list to choose the authentication type, either Windows Authentication or SQL Server Authentication (based on the authentication types selected when you installed the server). Provide a Windows user name or SQL Server login ID and password as necessary.
4. Click Options to display the advanced view of the Connect To Server dialog box. Select the Connection Properties tab, as shown in Figure 5-2.
5. In the Connect To Database box, type the name of the database to which you want to connect, such as Personnel. Or select Browse Server in the related drop-down list. When prompted, click Yes to establish a connection to the previously designated server. In the Browse Server For Database dialog box, select the database you want to use, and then click OK.
6. Using the Network Protocol list, select the network protocol and any other connection properties if you are prompted to do so. Shared Memory is the default network protocol for local connections. TCP/IP is the default for remote connections. Optionally, establish a secure connection by selecting Encrypt Connection.
7. Click Connect. You will then be able to work with the specified database in the Object Explorer view.

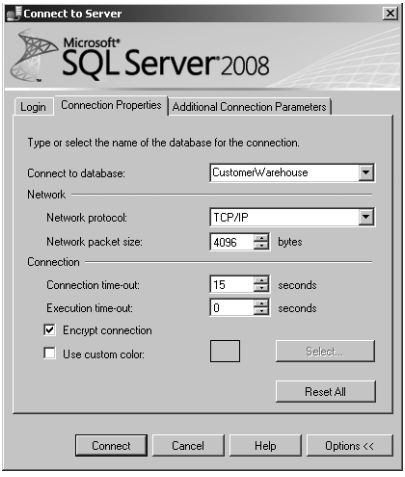

Figure 5-2 The Connection Properties tab in the Connect To Server dialog box.

Managing SQL Server Groups

You use SQL Server groups to organize sets of SQL servers. You define these server groups, and you can organize them by function, department, or any other criteria. Creating a server group is easy. You can even create subgroups within a group, and if you make a mistake, you can delete groups as well.

Introducing SQL Server Groups

In SQL Server Management Studio, you use the Registered Servers view to work with server groups. To use this view, or to display it if it is hidden, press Ctrl+Alt+G.

The top-level groups are already created for you, based on the SQL Server instances. Use the Registered Servers toolbar to switch between the various top-level groups. These top-level groups are organized by SQL Server instance:

- Database Engine
- Analysis Services
- Reporting Services
- SQL Server Compact Edition
- Integration Services

Although you can add registered servers directly to the top-level groups (as explained in "Managing Servers" on page 119), in a large enterprise with many SQL Server instances, you will probably want to create additional levels in the server group

hierarchy. These additional levels make it easier to access and work with your SQL servers. You can use the following types of organizational models:

- **Division or business unit model** In this model, group names reflect the divisions or business units to which the SQL servers belong or in which they are located. For example, you could have Engineering, IS, Operations, and Support server groups.
- **Geographic location model** In this model, group names reflect the geographic location of SQL servers, such as North America and Europe. You could have additional levels under North America for USA, Canada, and Mexico, for example, and levels under Europe could include UK, Germany, and Spain.

Figure 5-3 shows an example using server groups. As the figure shows, subgroups are organized under their primary group. Under Database Engine, you might have Corporate Customers, Engineering, and Enterprise Data groups. Within Engineering, you might have Dev, Test, and Core subgroups.

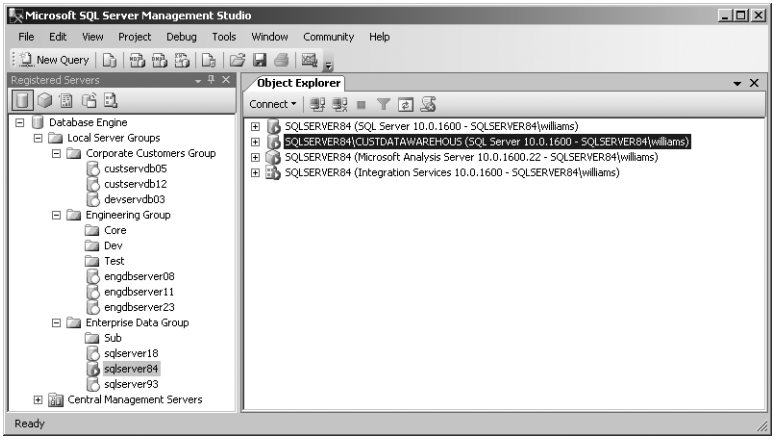

Figure 5-3 Use server groups to organize SQL Server deployments.

Creating a Server Group

You can create a server group or a subgroup by completing the following steps:

1. In SQL Server Management Studio display the Registered Servers view by pressing Ctrl+Alt+G. If the view was previously hidden, this also displays the view.
2. Use the Registered Servers toolbar to select the top-level group. For example, if you want to create a second-level or third-level group for Database Engine instances, select Database Engine.
3. As necessary, expand the top-level group node and the Local Server Groups nodes by double-clicking each in turn. You will see the names of the top-level

server group and any second-level server groups that you created. You can now do the following:

- Add the server group to one of the top-level or second-level groups by right-clicking the group name and choosing New Server Group.
- Add the server group to a lower-level group by expanding the server group entries until the group you want to use is displayed. Then right-click the group name and choose New Server Group.

4. In the New Server Group Properties dialog box, shown in Figure 5-4, type a name and description for the new group in the boxes provided. Click OK.

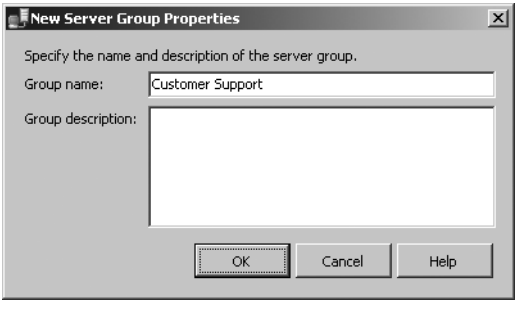

Figure 5-4 Enter a name and description in the New Server Group Properties dialog box.

Deleting a Server Group

You can delete a group or subgroup by completing the following steps:

1. In SQL Server Management Studio, display the Registered Servers view by pressing Ctrl+Alt+G. If the view was previously hidden, this also displays the view.
2. Use the Registered Servers toolbar to select the top-level group in which the group you want to delete is located. For example, if you want to delete a second-level or third-level group for Database Engine instances, select Database Engine.
3. Click the plus sign (+) next to the group or subgroup you want to delete. If the group has servers registered in it, move them to a different group. (The steps involved in moving servers to a new group are explained in "Moving a Server to a New Group" on page 126.)
4. Select the group or subgroup entry.
5. Press Delete. When prompted to confirm the action, click Yes.

Editing and Moving Server Groups

Server groups have several key properties you can edit: the name, the description, and the location in the Registered Server hierarchy. To edit a group's name or description, follow these steps:

1. Right-click the group in the Registered Servers view, and then select Properties.
2. In the Edit Server Group Properties dialog box, enter the new group name and description. Click OK.

To move a group (and all its associated subgroups and servers) to a new level in the server group hierarchy, follow these steps:

1. Right-click the group in the Registered Servers view, point to Tasks, and then select Move To.
2. In the Move Server Registration dialog box, you can now do the following:
   - ❑ Move the group to the top-level group by selecting the top-level group. This will make the group a second-level group.
   - ❑ Move the group to a different level by selecting a subgroup into which you want to place the group.
3. Click OK.

Adding SQL Servers to a Group

When you register a SQL Server for use with SQL Server Management Studio, you can choose the group in which you want the server. You can even create a new group specifically for the server. The next section, "Managing Servers," covers the topic of server registration.

Managing Servers

Servers and databases are the primary resources you manage in SQL Server Management Studio. When you select a top-level group in the Registered Servers view, you can see the available server groups. If you expand the view of these groups by double-clicking the group name, you can see the subgroups or servers assigned to a particular group. Local servers are registered automatically (in most cases). If a local server is not shown, you will need to update the local registration information. If the remote server you want to manage is not shown, you will need to register it.

Registration saves the current connection information and assigns the server to a group for easy future access using the Registered Servers view. Once you register a server, you can connect to the server to work with it and then disconnect when you are finished simply by double-clicking the server entry in the Registered Servers view. If you are not automatically connected, you can force a connection by right-clicking the server entry, and then selecting New Query (if you want to make a SQL query) or Object Explorer (if you want to view and manage the server).

You can start the registration process by using either of the following techniques:

- Register a server to which you are connected in Object Explorer.
- Register a new server in the Registered Servers view.

You can manage previous registrations in a variety of ways:

- Import registration information on previously registered SQL Server 2000 servers.
- Update registration information for local servers.
- Copy registration information from one computer to another using import and export.

Registering a Connected Server

Any server to which you have connected in Object Explorer can be registered easily. Registration saves the current connection information and assigns the server to a group for easy future access using the Registered Servers view. To register a connected server, follow these steps:

1. Right-click any server to which you are currently connected in the Object Explorer view, and then choose Register to display the New Server Registration dialog box shown in Figure 5-5.

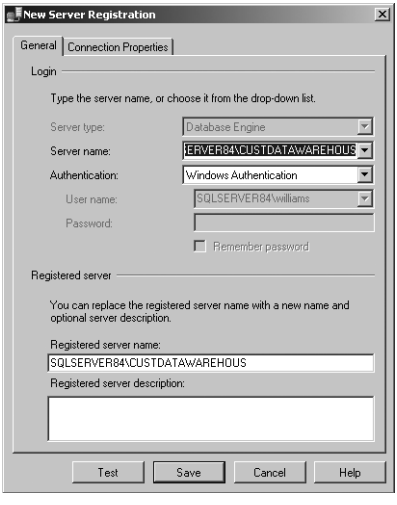

Figure 5-5 The New Server Registration dialog box.

2. On the General tab, the current values for the server name and authentication are filled in for you. Although the Registered Server Name option is set to the same value as the server name, you can modify this name and add a description if desired.

3. On the Connection Properties tab, you can specify the database to which you want to connect and set options for networking and connections. If you want to encrypt the connection, select the Encrypt Connection check box.
4. To test your settings before saving the registration settings, click Test. If the test was unsuccessful, verify the settings and then make changes as necessary. As discussed in Chapter 3, SQL Server doesn't allow remote connections by default and you must change the configuration settings to allow remote connections.
5. Click Save to save the server registration.

By default, the server is added to the top-level group. To move the server to a new level in the server group hierarchy, follow these steps:

1. Right-click the servers in the Registered Servers view, point to Tasks, and then select Move To.
2. In the Move Server Registration dialog box, you can move the server to a different level by selecting the subgroup into which you want to place the group.
3. Click OK.

Registering a New Server in the Registered Servers View

You do not have to connect to a server in Object Explorer to register it. You can register new servers directly in the Registered Servers view by following these steps:

1. In the Registered Servers view, use the toolbar to select the type of server to which you want to connect, such as Database Engine.
2. After you expand the available groups as necessary, right-click the group into which you want to place the server in the Registered Servers view and then select New Server Registration to display the New Server Registration dialog box shown previously in Figure 5-5.
3. In the Server Name box, type the fully qualified domain name or host name of the server on which SQL Server is running, such as **corpsvr04.cpandl.com** or **CorpSvr04**.
4. Use the Authentication list to choose the authentication type, either Windows Authentication or SQL Server Authentication (based on the authentication types selected when you installed the server). Provide a Windows user name or SQL Server login ID and password as necessary.
    - **Windows Authentication** Uses your current domain account and password to establish the database connection. This authentication type works only if Windows authentication is enabled and you have appropriate privileges.
    - **SQL Server Authentication** Allows you to specify a SQL Server login ID and password. To save the password so that you do not have to re-enter it each time you connect, select Remember Password.

5. You can also set connection settings using the options on the Connection Properties tab. These options allow you to connect to a specific database instance and to set the network configuration. If you want to encrypt the connection, select the Encrypt Connection check box.
6. The registered server name is filled in for you based on the previously entered server name. Change the default name only if you want SQL Server Management Studio to use an alternate display name for the server.
7. To test the settings, click Test. If you successfully connect to the server, you will see a prompt confirming this. If the test fails, verify the information you provided, make changes as necessary, and then test the settings again.
8. Click Save.

Registering Previously Registered SQL Server 2000 Servers

Registration details for servers registered by SQL Server 2000 can be imported into SQL Server Management Studio. This makes it easier to work with existing SQL Server 2000 installations. If the SQL Server 2000 installations were previously registered on the computer, you can import the registration details into a specific server group by completing the following steps:

1. In the Registered Servers view, use the toolbar to select the type of servers you are registering, such as Database Engine.
2. Right-click the Local Server Groups entry, point to Tasks, and then select Previously Registered Servers.
3. Available registration information for SQL Server 2000 servers will be imported. If an error prompt is displayed, you might not be logged on locally to the computer on which the servers were registered previously.

Updating Registration for Local Servers

Local servers are registered automatically (in most cases). If you have added or removed SQL Server instances on the local computer and those instances are not displayed, you will need to update the local server registration. Updating the registration information ensures that all currently configured local server instances are shown in SQL Server Management Studio.

To update registration details for local servers, follow these steps:

1. In the Registered Servers view, use the toolbar to select the type of servers you are registering, such as Database Engine.
2. Right-click the Local Server Groups entry, point to Tasks, and then select Register Local Servers.

Copying Server Groups and Registration Details from One Computer to Another

After you have registered servers in SQL Server Management Studio and placed the servers into a specific group hierarchy, you might find that you want to use the same registration information and server group structure on another computer. SQL Server Management Studio allows you to copy registration information from one computer to another using an import/export process. You can copy the registration details with or without the user names and passwords.

To export the registration and group information to a file on one computer and then import it onto another computer, complete the following steps:

1. Start SQL Server Management Studio on the computer with the registration and group structure details that you want to copy.
2. Select the Registered Servers view by pressing Ctrl+Alt+G.
3. In the Registered Servers view, use the toolbar to select the type of servers you want to work with, such as Database Engine.
4. Right-click the Local Server Groups entry, point to Tasks, and then select Export to display the Export Registered Servers dialog box shown in Figure 5-6.

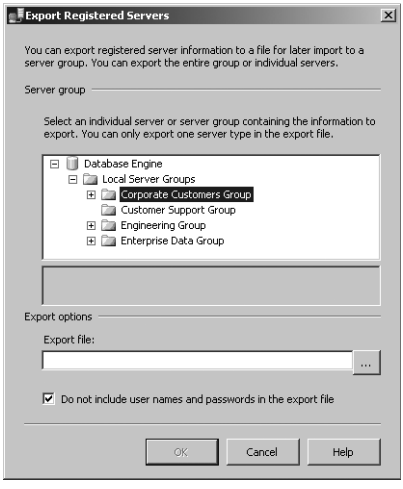

Figure 5-6 The Export Registered Servers dialog box.

5. Under Server Group, select the point from which the export process will begin. You can start copying registration information at any level in the group structure:
   - To copy the structure for a top-level group, all its subgroups, and all registration details for all related servers, select the Local Server Groups entry.

- To copy the structure for a subgroup, its subgroups (if any), and all registration details for all related servers, select a subgroup.
- To copy the registration details for a single server, select the server.

6. The server group structure and registration details are exported to a Registration Server File with the .regsrvr extension. By default, this file is created in the %*SystemRoot*%\System32 folder. Under Export Options, type a name for the registration server file, such as **CurrentDBConfig**.

   Tip If you place the registration server file on a secure network share, you can access it on the computer to which you want to copy the registration information. Otherwise, you will need to copy this file to this computer later.

7. By default, the current authentication details for server connections are not exported into the saved file. If you want to export user names and passwords, clear the Do Not Include User Names And Passwords In The Export File check box.
8. Click OK. If the export was successful, you will see a dialog box confirming this. Click OK in the dialog box. If there was a problem, note and correct the problem.
9. Start SQL Server Management Studio on the computer to which you want to copy the server group and registration details. If you did not place the registration server file on a secure network share, you will need to copy the file to this computer now.
10. Select the Registered Servers view by pressing Ctrl+Alt+G.
11. In the Registered Servers view, use the toolbar to select the type of servers you want to work with, such as Database Engine.
12. Right-click the Local Server Groups entry, point to Tasks, and then select Import to display the Import Registered Servers dialog box shown in Figure 5-7.
13. Click the button to the right of the Import File text box in the dialog box, and then use the Open dialog box that appears to select the registration server file you want to import.
14. Under Server Group, select the server group under which you want the imported groups and servers to be created.
15. Click OK. If the import was successful, you will see a dialog box confirming this. Click OK in the dialog box. If there was a problem, note and correct the problem.

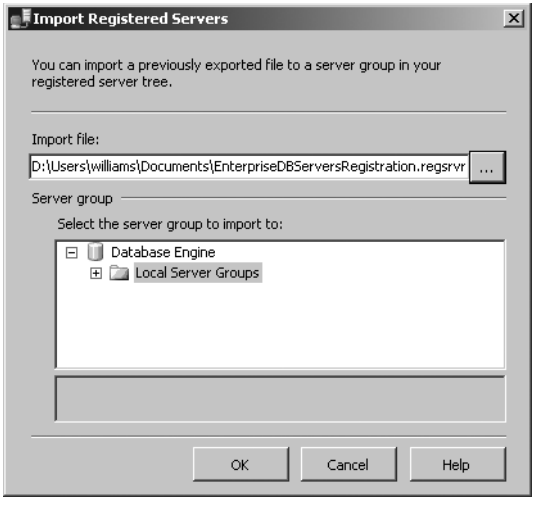

Figure 5-7 The Import Registered Servers dialog box.

Editing Registration Properties

You can change a server's registration properties at any time by right-clicking the server entry in the Registered Servers view in SQL Server Management Studio and then selecting Properties. Use the Edit Server Registration Properties dialog box to make necessary changes. The only property you cannot change is the server type. Be sure to test the settings before saving them.

Connecting to a Server

Once you have registered a server, connecting to it is easy. Right-click the server entry in the Registered Servers view in SQL Server Management Studio, and then select New Query (if you want to make an SQL query) or Object Explorer (if you want to view and manage the server). Or you can double-click the server entry to establish a connection, and then work with the server in the Object Explorer view.

> **Note** SQL Server Management Studio connects to other servers that are running SQL Server by using the network protocol set in the registration properties. If you have disabled the network protocol or remote access entirely for a server, however, you will not be able to connect to that server in SQL Server Management Studio. You will need to make the appropriate changes in the registration properties or in the surface area configuration. Chapter 3 discusses surface area configuration.

Disconnecting from a Server

When you are finished working with a server, you might want to disconnect from it. This cuts down on the back-and-forth communications to the server. To disconnect, right-click the server's entry in the Object Explorer view in SQL Server Management Studio, and then select Disconnect from the shortcut menu.

Moving a Server to a New Group

To move the server to a new group, complete the following steps:

1. Right-click the server in the Registered Servers view, point to Tasks, and then select Move To from the shortcut menu to display the Move Server Registration dialog box.
2. In the Move Server Registration dialog box, expand the Local Server Groups entry to see a list of subgroups. Expand subgroups as necessary. You can now do the following:
   - Move the server to the top-level group by selecting the top-level group. This will make the server a member of the top-level group.
   - Move the server to a different level by selecting a subgroup into which you want to place the server.
3. Click OK.

Deleting a Server Registration

If you change a server name or remove a server, you might want to delete the server registration in SQL Server Management Studio so that SQL Server Management Studio no longer tries to connect to a server that cannot be accessed. Right-click the server entry in the Registered Servers view, and then select Delete. When prompted to confirm the action, click Yes to delete the server registration details.

Starting, Stopping, and Configuring SQL Server Agent

SQL Server Agent runs as a service, and it is used to schedule jobs, alerts, and other automated tasks. After you have scheduled automated tasks, you will usually want SQL Server Agent to start automatically when the system starts. This configuration ensures that the scheduled tasks are performed as expected. Using SQL Server Service Manager, you can control the related SQL Server Agent (*InstanceName*) service just as you do the SQL Server service. For details, see "Configuring SQL Server Services" on page 56.

You use SQL Server Management Studio to configure SQL Server Agent. Chapter 16, "Database Automation and Maintenance," covers the agent configuration in detail, but the basic steps are as follows:

1. Connect to the Database Engine on the server you want to configure. You can do this in the Registered Servers view by double-clicking the server entry, or you can use the Object Explorer view. In the Object Explorer view, click Connect, and then select Database Engine to display the Connect To Server dialog box, which you can use to connect to the server.
2. Right-click the SQL Server Agent node, and then select Properties from the shortcut menu. You can now configure SQL Server Agent. Keep in mind that if the service is not running, you will need to start it before you can manage its properties.
3. The SQL Server Agent shortcut menu also lets you manage the SQL Server Agent service. Select Start, Stop, or Restart as appropriate.

Starting, Stopping, and Configuring Microsoft Distributed Transaction Coordinator

Microsoft Distributed Transaction Coordinator (DTC) is a transaction manager that makes it possible for client applications to work with multiple sources of data in one transaction.

When a distributed transaction spans two or more servers, the servers coordinate the management of the transaction using DTC. When a distributed transaction spans multiple databases on a single server, SQL Server manages the transaction internally.

SQL Server applications can call DTC directly to start an explicit distributed transaction. Distributed transactions can also be started implicitly by using one of the following methods:

- Calling stored procedures on remote servers running SQL Server
- Updating data on multiple OLE DB data sources
- Enlisting remote servers in a transaction

If you work with transactions under any of these scenarios, you will want to have DTC running on the server, and you will probably also want DTC to start automatically when the server starts. As with SQL Server itself, DTC runs as a service. This service is named Distributed Transaction Coordinator. Unlike the SQL Server service, only one instance of the MS DTC service runs on a computer, regardless of how many database server instances are available. This means that all instances of SQL Server running on a computer use the same transaction coordinator.

You can view the current state of Distributed Transaction Coordinator in SQL Server Management Studio by connecting to the server's Database Engine. In Object Explorer, expand the server and Management nodes. If the service is running, you will

see a green circle with a right-facing triangle in it (similar to a play button). If the service is stopped, you will see a red circle with a square in it (similar to a stop button). You can control the DTC service with Computer Management. Follow these steps:

1. Start Computer Management by clicking the Start button, pointing to All Programs, Administrative Tools, and then selecting Computer Management.
2. By default, you are connected to the local computer. To connect to a remote computer, right-click the Computer Management node and then select Connect To Another Computer. In the Select Computer dialog box, choose Another Computer, and then type the name of the computer. The name can be specified as a host name, such as **CorpSvr04**, or a fully qualified domain name, such as **corpsvr04.cpandl.com**.
3. Expand Services And Applications, and then select Services. Right-click Distributed Transaction Coordinator, and then choose Properties. You can now manage DTC.

Managing SQL Server Startup

The SQL Server Database Engine has two modes of operation. It can run as a command-line application (SQLServr.exe) or as a service. Use the command-line application when you need to troubleshoot problems or modify configuration settings in single-user mode. Other than that, you will normally run SQL Server as a service.

Enabling or Preventing Automatic SQL Server Startup

In Chapter 3, you learned that you can use SQL Server Configuration Manager to manage the SQL Server (MSSQLSERVER) service, related services for other Database Engine instances, and other SQL Server–related services. Any of these services can be configured for automatic startup or can be prevented from starting automatically. To enable or prevent automatic startup of a service, follow these steps:

1. Start SQL Server Configuration Manager using one of the following techniques:
   - Log on to the database server through a local or remote login, and then start SQL Server Configuration Manager by clicking the Start button; pointing to All Programs, Microsoft SQL Server 2008, Configuration Tools; and then selecting SQL Server Configuration Manager.
   - In SQL Server Management Studio, open the Registered Servers view by pressing Ctrl+Alt+G. Use the Registered Servers toolbar to select the top-level group, and then expand the group nodes by double-clicking them. Right-click the server entry and then select SQL Server Configuration Manager.

2. Select the SQL Server Services node. Right-click the SQL Server service that you want to start automatically, and then select Properties. You can now do the following:
   - ❑ **Enable automatic startup** On the Service tab, set the Start Mode to Automatic. If the server state is Stopped, click Start on the Log On tab to start the service.
   - ❑ **Prevent automatic startup** On the Service tab, set the Start Mode to Manual.
3. Click OK.

You can also use Computer Management to configure services. To configure automatic startup of a service using Computer Management, follow these steps:

1. Click the Start button, point to All Programs, and then select Administrative Tools, Computer Management.
2. By default, you are connected to the local computer. To connect to a remote computer, right-click the Computer Management node and select Connect To Another Computer. In the Select Computer dialog box, select Another Computer and then type the name of the computer. The name can be specified as a host name, such as **CorpSvr04**, or a fully qualified domain name, such as **corpsvr04.cpandl.com**.
3. Expand Services And Applications, and then select Services.
4. Right-click the SQL Server service that you want to start automatically, and then select Properties.
5. You can now do the following:
   - ❑ **Enable automatic startup** On the General tab, set the Startup Type to Automatic. If the Service Status reads Stopped, click Start.
   - ❑ **Prevent automatic startup** On the General tab, set the Startup Type to Manual.
6. Click OK.

Setting Database Engine Startup Parameters

Startup parameters control how the SQL Server Database Engine starts and which options are set when it does. You can configure startup options using SQL Server Configuration Manager or Computer Management. SQL Server Configuration Manager is the recommended tool for this task because it provides the current default settings and allows you to easily make modifications.

Tip You can pass startup parameters to the command-line utility SQLServr.exe as well. Passing the –c option to this utility starts SQL Server without using a service. You must run SQLServr.exe from the Binn directory that corresponds to the instance of the SQL Server Database Engine that you want to start. For the default instance, the utility is located in MSSQL10.MSSQLSERVER\MSSQL\Binn. For named instances, the utility is located in MSSQL10.*InstanceName*\MSSQL\Binn.

Adding Startup Parameters

You can add startup parameters by completing the following steps:

1. Start SQL Server Configuration Manager using one of the following techniques:
   - Log on to the database server through a local or remote login, and then start SQL Server Configuration Manager by clicking the Start button; pointing to All Programs, Microsoft SQL Server 2008, Configuration Tools; and then selecting SQL Server Configuration Manager.
   - In SQL Server Management Studio, open the Registered Servers view by pressing Ctrl+Alt+G. Use the Registered Servers toolbar to select the top-level group, and then expand the group nodes by double-clicking them. Right-click the server entry and then select SQL Server Configuration Manager.
2. Select the SQL Server Services node. Right-click the SQL Server service that you want to modify, and then select Properties.
3. On the Advanced tab, click in the Startup Parameters box, and then press End to go to the end of the currently entered parameters. The –d, –e, and –l parameters are set by default. Be careful not to modify these or other existing parameters accidentally.
4. Each parameter is separated by a semicolon. Type a semicolon and then a hyphen followed by the letter and value of the parameter you are adding, such as **;–g512**.
5. Repeat step 3 and step 4 as necessary to specify additional parameters and values.
6. Click Apply to save the changes. The parameters are applied the next time the SQL Server instance is started. To apply the parameters right away, you must stop and then start the service by clicking Restart on the Log On tab.

Removing Startup Parameters

You can remove startup parameters by completing the following steps:

1. Start SQL Server Configuration Manager using one of the following techniques:
   - Log on to the database server through a local or remote login, and then start SQL Server Configuration Manager by clicking the Start button;

pointing to All Programs, Microsoft SQL Server 2008, Configuration Tools; and then selecting SQL Server Configuration Manager.

- In SQL Server Management Studio, open the Registered Servers view by pressing Ctrl+Alt+G. Use the Registered Servers toolbar to select the top-level group, and then expand the group nodes by double-clicking them. Right-click the server entry and then select SQL Server Configuration Manager.

2. Select the SQL Server Services node. Right-click the SQL Server service that you want to modify, and then select Properties.
3. On the Advanced tab, click in the Startup Parameters box. Each parameter is specified with a hyphen, parameter letter, and parameter value. A semicolon is used to separate parameter values, as shown in the following example:

   ```
   -g512;
   ```

4. Remove the parameter by deleting its related parameter entry.
5. The change is applied the next time the SQL Server instance is started. To apply the parameters right away, you must stop and then start the service by clicking Restart on the Log On tab.

Common Startup Parameters

Table 5-1 shows startup parameters and how they are used. The first three parameters (−d, −e, and −l) are the defaults for SQL Server. The remaining parameters allow you to configure additional settings.

Table 5-1 Startup Parameters for SQL Server

Parameter	Description
–d<*path*>	Sets the full path for the *master* database. If this parameter is omitted, the registry values are used. **Example**: –dC:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\master.mdf
–e<*path*>	Sets the full path for the error log. If this parameter is omitted, the registry values are used. **Example**: –eC:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\LOG\ERRORLOG
–l<*path*>	Sets the full path for the *master* database transaction log. If this parameter is omitted, the registry values are used. **Example**: –lC:\Program Files\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\DATA\mastlog.ldf
–B	Sets a breakpoint on error; used with the –y option when debugging.
–c	Prevents SQL Server from running as a service. This setting makes startup faster when you are running SQL Server from the command line.

Table 5-1 Startup Parameters for SQL Server

Parameter	Description
–f	Starts SQL Server with minimal configuration. This setting is useful if a configuration value has prevented SQL Server from starting.
–g *number*	Specifies the amount of virtual address space memory in megabytes to reserve for SQL Server. This memory is outside the SQL Server memory pool and is used by the extended procedure .dlls, OLE DB providers referenced in distributed queries, and the automation object referenced in Transact-SQL (T-SQL). The default value is 256. **Example**: –g256
–h	Reserves virtual address space for Hot-Add Memory metadata when Address Windowing Extensions (AWE) is enabled with 32-bit editions of SQL Server. Although this is required for using Hot-Add Memory with 32-bit AWE, it uses approximately 500 MB of virtual address space and makes memory tuning more difficult. This setting is not required for using Hot-Add Memory with 64-bit editions of SQL Server. Hot-Add Memory is available only for Enterprise and Datacenter Editions of Windows Server and must also be supported by the server hardware.
–K	Forces regeneration of the service master key if it exists.
–k *number*	Sets the checkpoint speed in MB per second. Use a decimal value. **Example**: –k25
–m	Starts SQL Server in single-user mode. Only a single user can connect, and the checkpoint process is not started. Enables the *Sp_Configure* Allow Updates option, which is disabled by default.
–n	Tells SQL Server not to log errors in the application event log. Use with –e.
–s *instance*	Starts the named instance of SQL Server. You must be in the relevant Binn directory for the instance. **Example**: –sdevapps
–T<*tnum*>	Sets a trace flag. Trace flags set nonstandard behavior and are often used in debugging or diagnosing performance issues. **Example**: –T237
–t<*tnum*>	Sets an internal trace flag for SQL Server. Used only by SQL Server support engineers. **Example**: –t8837
–x	Disables statistics tracking for CPU time and cache-hit ratio. Allows maximum performance.
–y *number*	Sets an error number that causes SQL Server to dump the stack. **Example**: –y1803

Managing Services from the Command Line

You can start, stop, and pause SQL Server as you would any other service. On a local system, you can type the necessary command at a standard command prompt. On a remote system, you can connect to the system remotely and then issue the necessary command. To manage the default database server instance, use these commands:

- **NET START MSSQLSERVER** Starts SQL Server as a service.
- **NET STOP MSSQLSERVER** Stops SQL Server when running as a service.
- **NET PAUSE MSSQLSERVER** Pauses SQL Server when running as a service.
- **NET CONTINUE MSSQLSERVER** Resumes SQL Server when running as a service.

To manage named instances of SQL Server, use the following commands:

- **NET START MSSQL$*instancename*** Starts SQL Server as a service, where *instancename* is the actual name of the database server instance.
- **NET STOP MSSQL$*instancename*** Stops SQL Server when running as a service, where *instancename* is the actual name of the database server instance.
- **NET PAUSE MSSQL$*instancename*** Pauses SQL Server when running as a service, where *instancename* is the actual name of the database server instance.
- **NET CONTINUE MSSQL$*instancename*** Resumes SQL Server when running as a service, where *instancename* is the actual name of the database server instance.

You can add startup options to the end of net start MSSQLSERVER or net start MSSQL$*instancename* commands. Use a slash (/) instead of a hyphen (–) as shown in these examples:

```
net start MSSQLSERVER /f /m
net start MSSQL$CUSTDATAWAREHOUS /f /m
```

> **Real World** Instead of referencing MSSQLSERVER or MSSQL$*instancename*, you also can reference the service by its display name. For the default instance, you use "SQL Server (MSSQLSERVER)" with net start, net stop, net pause, and net continue. For a named instance, you use net start "SQL Server (*InstanceName*)", where *InstanceName* is the name of the instance, such as net start "SQL Server (CUSTDATAWAREHOUS)". In both usages, the quotation marks are required as part of the command text.

Managing the SQL Server Command-Line Executable File

The SQL Server command-line executable file (SQLServr.exe) provides an alternative to the SQL Server service. You must run SQLServr.exe from the Binn directory that corresponds to the instance of the SQL Server Database Engine that you want to start. For

the default instance, the utility is located in MSSQL10.MSSQLSERVER\MSSQL\Binn. For named instances, the utility is located in MSSQL10.*InstanceName*\MSSQL\Binn.

When SQL Server is installed on a local system, start SQL Server by changing to the directory where the instance of SQL Server you want to start is located, and then type **sqlservr** at the command line. On a remote system, connect to the system remotely, change to the appropriate directory, and then issue the startup command. Either way, SQL Server reads the default startup parameters from the registry and starts execution.

You can also enter startup parameters and switches that override the default settings. (The available parameters were summarized in Table 5-1.) You can still connect SQL Server Management Studio and SQL Configuration Manager to the server. However, when you do, these programs will show an icon indicating that the SQL Server service is stopped because you aren't running SQL Server via the related service. You also will be unable to pause, stop, or resume the instance of SQL Server as a Microsoft Windows service.

When you are running SQL Server from the command line, SQL Server runs in the security context of the user, not the security context of the account assigned to the SQL Server service. You should not minimize the command console in which SQL Server is running because doing so will cause Windows to remove nearly all resources from SQL Server.

Additionally, when you are running SQL Server from the command line, you can make configuration changes that might be necessary for diagnosing and resolving problems as well as tasks that you can accomplish only when SQL Server is running in single-user mode. However, you should be careful when creating databases, changing data file locations, or making other similar types of changes. If you are logged in as an administrator and create a new database or change the location of a data file, SQL Server might not be able to access the database or data file when running later under the default account for the SQL Server service.

You must shut down the instance of SQL Server before logging off Windows. To stop an instance of SQL Server started from the command line, complete the following steps:

1. Press Ctrl+C to break into the execution stream.
2. When prompted, press Y to stop SQL Server.

Managing Server Activity

As a database administrator, it is your job to make sure that SQL Server runs smoothly. To ensure that SQL Server is running optimally, you can actively monitor the server to

- Keep track of user connections and locks
- View processes and commands that active users are running
- Check the status of locks on processes and objects

- See blocked or blocking transactions
- Ensure that processes complete successfully, and detect errors if they do not

When problems arise, you can terminate a process, if necessary.

> **Note** For more coverage of monitoring SQL Server, see Chapter 14. In that chapter, you will learn how to use Performance Monitor and SQL Server Profiler to keep track of SQL Server activity, performance, and errors.

Examining Process Information

Process information provides detailed information about the status of processes, current user connections, and other server activity. You can view process information by completing the following steps:

1. Start SQL Server Management Studio, and then connect to a server.
2. Use the Object Explorer view to access an instance of the Database Engine.
3. Right-click the Database Engine instance and then select Activity Monitor.

In Activity Monitor, shown in Figure 5-8, you should see a graphical overview of activity as well as an activity summary by processes, resource waits, data file I/O, and recent expensive queries. The overview and the summaries are provided in separate panels that you can expand to display or shrink to hide.

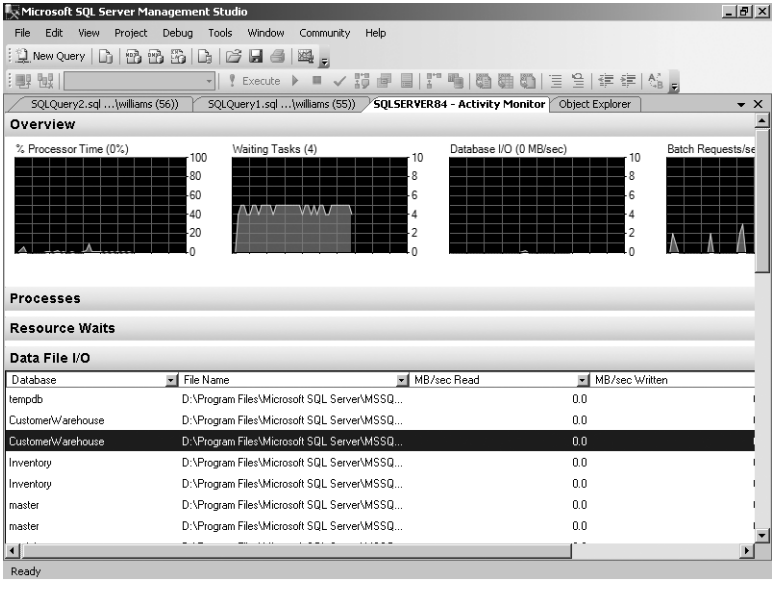

Figure 5-8 Working with Activity Monitor.

The Overview panel has graphs depicting processor time, waiting tasks, database I/O, and batch requests. The graphs are updated automatically every 10 seconds by default. You can specify a different refresh interval by right-clicking in the panel, pointing to Refresh Interval, and then selecting the desired interval, such as 30 seconds.

In the Processes panel, processes are sorted by process ID by default, but you can arrange them by any of the available information categories summarized in Table 5-2. Click a category header to sort processes based on that category. Click the same category header again to perform a reverse sort on the category.

Table 5-2 Process Information Used in Database Administration

Category	Description
Session ID	Provides the session ID of the process on the server.
User Process	Provides the user ID of the process on the server.
Login	Shows which user is running the process by SQL Server ID or domain account, depending on the authentication technique used.
Database	Indicates the database with which the process is associated.
Task Status	Shows the status of the process. A running process is active and currently performing work. A runnable process has a connection but currently has no work to perform. A sleeping process is waiting for something such as user input or a lock. A background process is running in the background and periodically performing tasks. A suspended process has work to perform but has stopped.
Command	Displays the command being executed or the last command executed.
Application	Shows the application or SQL Server component connecting to the server and running the process, such as a report server.
Wait Time	Indicates the elapsed wait time in milliseconds.
Wait Type	Specifies whether the process is waiting or not waiting.
Wait Resource	Displays the resource that the process is waiting for (if any).
Blocked By	Displays the process ID blocking this process.
Head Blocker	Shows 1 if the session ID is the head blocker in the blocking chain. Otherwise shows 0.
Memory Use	Displays the amount of memory the process is using (in KB).
Host Name	Displays the host from which the connection originated.
Workload Group	Displays the name of the Resource Governor workload group for the query.

Tracking Resource Waits and Blocks

When you are diagnosing performance issues, you'll want to look closely at the Wait Time, Wait Type, Wait Resource, and Blocked By values for each process. Most of the process information is gathered from data columns returned by various dynamic management views, including the following:

- **sys.dm_os_tasks** Returns information about each task that is active in the instance of SQL Server.
- **sys.dm_os_waiting_tasks** Returns information about each task that is waiting on some resource.
- **sys.dm_exec_requests** Returns information about each request that is executing within SQL Server.
- **sys.dm_exec_sessions** Returns information about each authentication session within SQL Server.
- **sys.dm_resource_governor_workload_group** Returns information about workload groups configured for Resource Governor.

Although Activity Monitor provides a good overview, you might need to use these dynamic management views to get more detailed information about processes, resource waits, and resource blocks.

The Resource Waits panel provides additional information about resource waits. Each wait category combines the wait time for closely related wait types, such as buffer I/O or network I/O. Keep the following in mind:

- **Wait Time** Shows the accumulated wait time per second. Here, a rate of 3000 ms per second indicates three tasks on average were waiting with this wait category.
- **Recent Wait Time** Shows the wait average of accumulated wait time per second. This combines all the wait times over the last several minutes and averages them for this wait category.
- **Average Waiter Count** Shows the average number of waiting tasks per second for this wait category.
- **Cumulative Wait Time** Shows the total amount of wait time for this wait category since SQL Server was started or the wait statistics were reset.

Tip You can reset wait statistics using DBCC SQLPERF.

To get a clearer picture of resource waits and blocks, you can use the sys.dm_tran_locks view. Table 5-3 summarizes the information returned with this view with actual values in parentheses preceded by a general category name.

Table 5-3 Lock-Related Information Used in Database Administration

Category	Type	Description
Process ID (request_session_id)		The process ID of the related user process within SQL Server.
Object ID (resource_associated_entity_id)		The ID of the entity with which a resource is associated.
Context (request_exec_context_id)		The ID of the thread associated with the process ID.
Batch ID (request_request_id)		The batch ID associated with the process ID.
Type (resource_type)	RID	Row identifier; used to lock a single row within a table.
	KEY	A row lock within an index; used to protect key ranges.
	PAGE	A lock on a data or index page.
	EXTENT	A lock on a contiguous group of eight data or index pages.
	TABLE	A lock on an entire table, including all data and indexes.
	DATABASE	A lock on an entire database.
	METADATA	A lock on descriptive information about the object.
	ALLOCATION_UNIT	A lock on allocation unit page count statistics during deferred drop operations.
	HOBT	A lock on basic access path structures for heap or index reorganization operations or heap-optimized bulk loads.
Subtype (resource_subtype)		The lock subtype, frequently used with METADATA locks to identify metadata lock activity.
Description (resource_description)		Optional descriptive information.

Table 5-3 Lock-Related Information Used in Database Administration

Category	Type	Description
Request Mode (request_mode)	S	Shared; used for read-only operations, such as a *select* statement.
	U	Update; used when reading/locking an updatable resource; prevents some deadlock situations.
	X	Exclusive; allows only one session to update the data; used with the modification operations, such as INSERT, DELETE, and UPDATE.
	I	Intent; used to establish a lock hierarchy.
	Sch-S	Schema stability; used when checking a table's schema.
	Sch-M	Schema modification; used when modifying a table's schema.
	BU	Bulk update; used when bulk copying data into a table and the TABLOCK hint is specified.
	RangeS_S	Serializable range scan; used with shared resource locks on shared ranges.
	RangeS_U	Serializable update; used for updating resource locks on shared ranges.
	RangeI_N	Insert range with a null resource lock; used to test ranges before inserting a new key into an index.
	RangeX_X	Exclusive range with an exclusive lock; used when updating a key in a range.
Request Type (request_type)		The type of object requested.

Table 5-3 Lock-Related Information Used in Database Administration

Category	Type	Description
Request Status (request_status)	GRANT	The lock was obtained.
	WAIT	The lock is blocked by another process.
	CNVT	The lock is being converted—that is, it is held in one mode but waiting to acquire a stronger lock mode.
Owner Type (request_owner_type)	CURSOR	The lock owner is a cursor.
	SESSION	The lock owner is a user session.
	TRANSACTION	The lock owner is a transaction.
	SHARED_TRANSACTION_WORKSPACE	The lock owner is the shared portion of the transaction workspace.
	EXCLUSIVE_TRANSACTION_WORKSPACE	The lock owner is the exclusive portion of the transaction workspace.
Owner ID (request_owner_id)		The owner ID associated with the lock.
Owner GUID (request_owner_guid)		The GUID of the owner associated with the lock.
Database (resource_database_id)		The database containing the lock.
Object (resource_associated_entity_id)		The name of the object being locked.

Troubleshooting Deadlocks and Blocking Connections

Two common problems you might encounter are deadlocks and blocking connections. Deadlocks and blocking connections, as described in the following list, can occur in almost any database environment, especially when many users are making connections to databases:

- Deadlocks occur when two users have locks on separate objects and each wants a lock on the other's object. Each user waits for the other user to release the lock, but this does not happen.
- Blocking connections occur when one connection holds a lock and a second connection wants a conflicting lock type. This forces the second connection either to wait or to block the first.

Both deadlocks and blocking connections can degrade server performance.

Although SQL Server can detect and correct deadlock and blocking situations, you can help speed up this process by identifying potential problems and taking action, if necessary. Process information can tell you when deadlocks or blocking occur. Examine these process information columns: Wait Time, Wait Type, Resource, Blocking, and Blocked By. When you have a deadlock or blocking situation, take a closer look at the locks on the objects that are causing problems. Refer to "Tracking Resource Waits and Blocks" on page 137 for details. You might also want to stop the offending processes, and you can do this by following the steps described in "Killing Server Processes" on page 143.

You can also use the sys.dm_tran_locks view to obtain information about active locks. Each row in the results returned by this view represent a currently active request to the lock manager for a lock that has been granted or is waiting to be granted. The following example returns a list of locks in the Customer database:

```
USE customer;
GO
SELECT * FROM sys.dm_tran_locks
```

In the result set, the results are organized in two main groups. Columns that begin with *resource_* describe the resource on which the lock request is being made. Columns that begin with *request_* describe the lock request itself. Earlier, Table 5-3 listed the correlation between the columns in the results and the categories listed in Activity Monitor. While Activity Monitor returns the actual database name, the resource_database_id column returns the database_id as set in the sys.databases view. In SQL Server, database IDs are set on a per-server basis. You can determine the database name for a particular database ID on a particular server using the following statement:

```
SELECT name, database_id FROM sys.databases
```

In the results returned by sys.dm_tran_locks, the request_session_id tracks process IDs. Process IDs as tracked internally by SQL Server do not correspond to process IDs tracked by the operating system. You can determine the association between the SQL Server process IDs and Windows thread IDs using the following query:

```
SELECT ServerTasks.session_id, ServerThreads.os_thread_id
    FROM sys.dm_os_tasks AS ServerTasks
    INNER JOIN sys.dm_os_threads AS ServerThreads
        ON ServerTasks.worker_address = ServerThreads.worker_address
    WHERE ServerTasks.session_id IS NOT NULL
    ORDER BY ServerTasks.session_id;
GO
```

While you are connected to the database that contains the locking object, you get more information about the locking object and blocking information. Use the following

query, where *<resource_associated_entity_id>* is the value in the related column, to get information about the locking object:

```
SELECT object_name(object_id), *
    FROM sys.partitions
    WHERE hobt_id=<resource_associated_entity_id>
```

Use the following query to get blocking information:

```
SELECT
        tr1.resource_type,
        tr1.resource_subtype,
        tr1.resource_database_id,
        tr1.resource_associated_entity_id,
        tr1.request_mode,
        tr1.request_type,
        tr1.request_status,
        tr1.request_session_id,
        tr1.request_owner_type,
        tr2.blocking_session_id
    FROM sys.dm_tran_locks as tr1
    INNER JOIN sys.dm_os_waiting_tasks as tr2
        ON tr1.lock_owner_address = tr2.resource_address;
```

Tracking Command Execution in SQL Server

Sometimes you will want to track the commands that users are executing. You can do this by using Activity Monitor:

1. In SQL Server Management Studio, use the Object Explorer view to access an instance of the Database Engine.
2. Right-click the Database Engine instance and then select Activity Monitor.
3. Expand the Processes panel by clicking the Options button. The entries in the Session ID, User Process, and Login columns can help you track user sessions and the processes they are using.
4. Right-click a process and then select Details to display the dialog box shown in Figure 5-9. This dialog box shows the last command batch executed by the user.
5. To track current commands being executed by the user, click Refresh periodically.
6. To kill the process, click Kill Process. Then, when prompted, choose Yes.

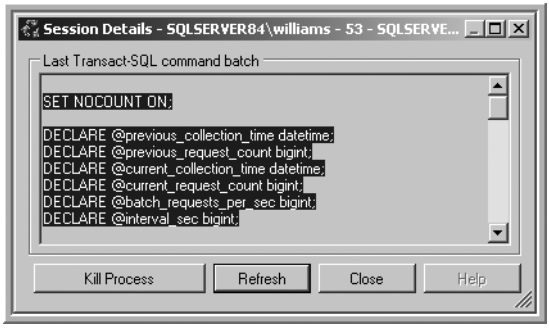

Figure 5-9 The Session Details dialog box.

Killing Server Processes

You might need to stop processes that are blocking connections or using too much CPU time. To do this, complete the following steps:

1. In SQL Server Management Studio, use the Object Explorer view to access an instance of the Database Engine.
2. Right-click the Database Engine instance and then select Activity Monitor.
3. Expand the Processes panel by clicking the Options button.
4. Right-click the process you want to stop and then choose Kill Process. When prompted to confirm, click Yes.

> **Note** Usually, you will not want to kill processes that SQL Server is running. If you are concerned about a process, stop it and then restart the related service instead of trying to kill the process.